Remarks

Prior to examination, the applicant requests entry of the amendments noted above. Claims 1-32 and a duplicate claim 28 were pending prior to this Amendment.

In the specification, numerous typographical errors were corrected. No new matter was introduced through these corrections.

In the claims section of the original application, claims 2 and 28 were cancelled. Claim 28 was inadvertently listed twice. Claims 11 and 20 were amended to better claim that which the applicant considers his invention. Claim 22 was amended to correct a typographical error. Claims 1-10, 12-19, 21 and 23-32 remain unchanged.

In the drawings, a number of replacement sheets are submitted to correct typographical errors or to enhance the readability of the figures. All of the drawing figure changes, with the exception of Figure 1E and 3, have been amended to look as they did in United States Provisional Patent Application Nos. 60/399,635 entitled "Data Dispersion and Mirroring Method with Fast Dual Erasure Correction and Multi-Dimensional Capabilities" filed on July 29, 2002, and 60/460,545 entitled "Composite Data Protection Method for Micro Level Data" filed April 4, 2003. Since these applications were incorporated into the present application by reference, no new matter has been added through these changes.

With respect to Figure 1E, the changes are supported by at least the text in paragraphs 0077 through 0080.

With respect to Figure 3, the changes are supported by at least the text in paragraph 0062.

If any fee is required for entry of this paper, the Commissioner is authorized to charge Deposit Account No. 06-0029 and is requested to notify us of the same.

Respectfully Submitted,

ROBERT HALFORD

Date: January 13, 2005

Robert B. Leonard, #33,946
FAEGRE & BENSON LLP
2200 Wells Fargo Center
90 South Seventh Street
Minneapolis, MN 55402-3901
Telephone: (612) 766-8578

M2:20572133.04

AMENDMENTS TO THE DRAWINGS

Please replace Figures 1D, 1E, 2A, 2C, 3, 5, 7, 12, 13, 14 and 17, with the attached Replacement Sheets. Annotated marked-up drawings are also provided that identify the changes in the above noted Figures.

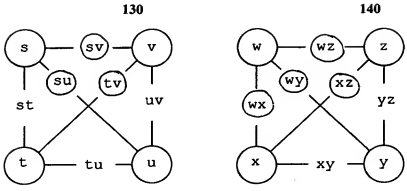

130
140
s
sv
v
su
tv
st
uv
t
tu
u
w
wz
z
wy
xz
wx
yz
x
xy
y

FIG 1D

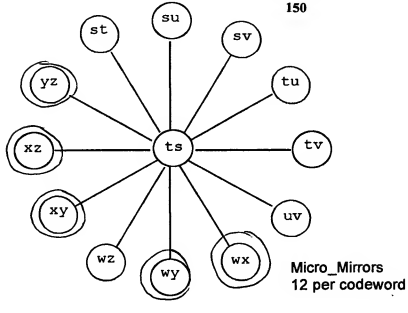

150
st
su
sv
yz
tu
xz
ts
tv
xy
uv
wz
wy
wx
Micro_Mirrors
12 per codeword

FIG 1E

200

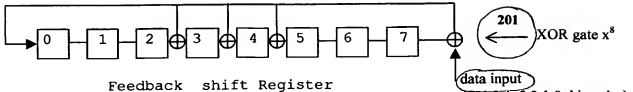


FIG 2A

0	0	0	0	0	0	0	0	
1	0	0	1	1	1	0	0	results after 1 shift
0	1	0	0	1	1	1	0	results after 2 shifts
0	0	1	0	0	1	1	1	results after 3 shifts
0	0	0	1	0	0	1	1	results after 4 shifts
0	0	0	0	1	0	0	1	results after 5 shifts
0	0	0	0	0	1	0	0	results after 6 shifts
0	0	0	0	0	0	1	0	results after 7 shifts
0	0	0	0	0	0	0	1	results after 8 shifts, ECC = 80

FIG 2B

210

e0 = d0 + d3 + d4 + d5 + d6
e1 = d1 + d4 + d5 + d6 + d7
e2 = d2 + d5 + d6 + d7
e3 = d0 + d4 + d5 + d7
e4 = d0 + d1 + d3 + d4
e5 = d0 + d1 + d2 + d3 + d6
e6 = d1 + d2 + d3 + d4 + d7
e7 = d2 + d3 + d4 + d5

per example

0 + 1 + 1 + 0 + 0	= 0
0 + 1 + 0 + 0 + 1	= 0
1 + 0 + 0 + 1	= 0
0 + 1 + 0 + 1	= 0
0 + 0 + 1 + 1	= 0
0 + 0 + 1 + 1 + 0	= 0
0 + 1 + 1 + 1 + 1	= 0
1 + 1 + 1 + 0	= 1

ECC = 80 for data byte 9C

FIG 2C

220

d0 = e2 + e3 + e4 + e5
d1 = e0+ e3 + e4 + e5 + e6
d2 = e1 + e4 + e5 + e6 + e7
d3 = e3 + e4 + e6 + e7
d4 = e0 + e2 + e3 + e7
d5 = e0 + e1 + e2 + e5
d6 = e0 + e1 + e2 + e3 + e6
d7 = e1 + e2 + e3 + e4 + e7

per example

0 + 0 + 0 + 0	= 0
0 + 0 + 0 + 0 + 0	= 0
0 + 0 + 0 + 0 + 1	= 1
0 + 0 + 0 + 1	= 1
0 + 0 + 0 + 1	= 1
0 + 0 + 0 + 0	= 0
0 + 0 + 0 + 0 + 0	= 0
0 + 0 + 0 + 0 + 1	= 1

data byte = 9C for ECC = 80

FIG 2D

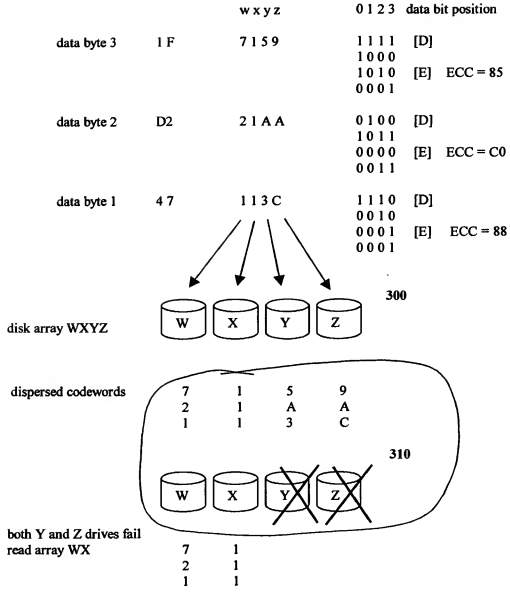
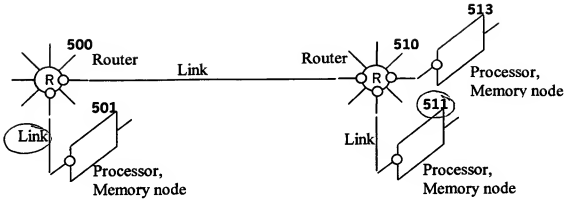

w x y z
0 1 2 3 data bit position
data byte 3
1 F
7 1 5 9
1 1 1 1 [D]
1 0 0 0
1 0 1 0 [E] ECC = 85
0 0 0 1
data byte 2
D2
2 1 A A
0 1 0 0 [D]
1 0 1 1
0 0 0 0 [E] ECC = C0
0 0 1 1
data byte 1
4 7
1 1 3 C
1 1 1 0 [D]
0 0 1 0
0 0 0 1 [E] ECC = 88
0 0 0 1
300
W
X
Y
Z
disk array WXYZ
dispersed codewords
7 1 5 9
2 1 A A
1 1 3 C
310
W
X
Y
Z
both Y and Z drives fail
read array WX
7 1
2 1
1 1
Recovered Data
1F
D2
47

FIG 3



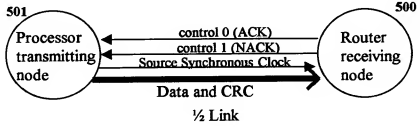


FIG 5

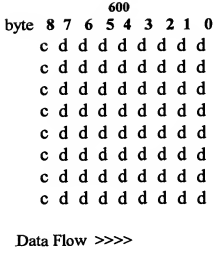


FIG 6A

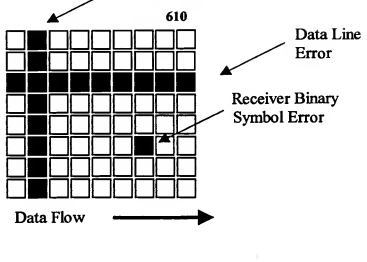


FIG 6B

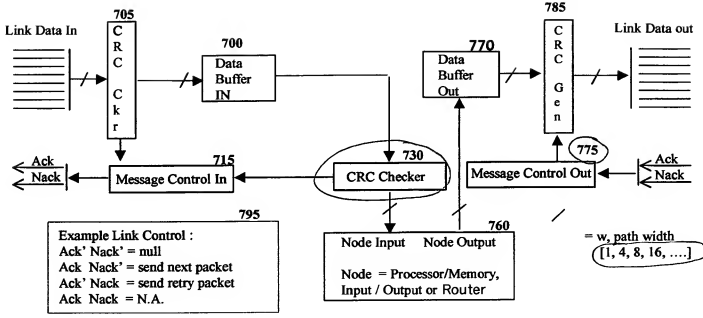

Link Data In
705
CRC Ckr
700
Data Buffer IN
770
Data Buffer Out
785
CRC Gen
Link Data out
Ack Nack
715
Message Control In
730
CRC Checker
775
Message Control Out
Ack Nack
795
Example Link Control :
Ack' Nack' = null
Ack Nack' = send next packet
Ack' Nack = send retry packet
Ack Nack = N.A.
760
Node Input Node Output
Node = Processor/Memory,
Input / Output or Router
= w, path width
(1, 4, 8, 16, ...)

FIG 7

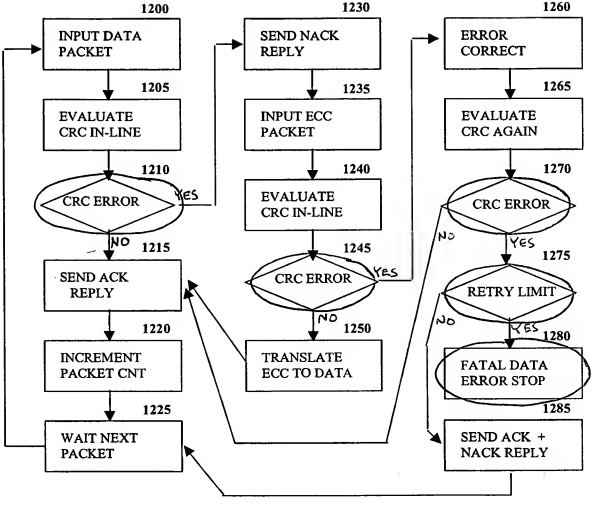

1200
INPUT DATA PACKET
1205
EVALUATE CRC IN-LINE
1210
CRC ERROR
YES
NO
1215
SEND ACK REPLY
1220
INCREMENT PACKET CNT
1225
WAIT NEXT PACKET
1230
SEND NACK REPLY
1235
INPUT ECC PACKET
1240
EVALUATE CRC IN-LINE
1245
CRC ERROR
YES
NO
1250
TRANSLATE ECC TO DATA
1260
ERROR CORRECT
1265
EVALUATE CRC AGAIN
1270
CRC ERROR
NO
YES
1275
RETRY LIMIT
NO
YES
1280
FATAL DATA ERROR STOP
1285
SEND ACK + NACK REPLY

FIG 12

[E]							1301	[D]							1300
c6uc2u	u30	u26	u22	u18	u14	u10	u06	u02	c4sc0s	s28	s24	s20	s16	s12	s08	s04	s00
c6vc2v	v30	v26	v22	v18	v14	v10	v06	v02	c4tc0t	t28	t24	t20	t16	t12	t08	t04	t00
c7uc3u	u31	u27	u23	u19	u15	u11	u07	u03	c5sc1s	s29	s25	s21	s17	s13	s09	s05	s01
c7vc3v	v31	v27	v23	v19	v15	v11	v07	v03	c5tc1t	t29	t25	t21	t17	t13	t09	t05	t01
c4uc0u	u28	u24	u20	u16	u12	u08	u04	u00	c6sc2s	s30	s26	s22	s18	s14	s10	s06	s02
c4vc0v	v28	v24	v20	v16	v12	v08	v04	v00	c6tc2t	t30	t26	t22	t18	t14	t10	t06	t02
c5uc1u	u29	u25	u21	u17	u13	u09	u05	u01	c7sc3s	s31	s27	s23	s19	s15	s11	s07	s03
c5vc1v	v29	v25	v21	v17	v13	v09	v05	v01	c7tc3t	t31	t27	t23	t19	t15	t11	t07	t03

Data Flow >>> <<< NACK <<< ACK

FIG 13

[E]							1401	[D]							1400
c6uc2u	u30	u26	u22	u18	u14	u10	u06	u02	c4sc0s	s28	s24	s20	s16	s12	s08	s04	s00
c6vc2v	v30	v26	v22	v18	v14	v10	v06	v02	c4tc0t	t28	t24	t20	t16	t12	t08	t04	t00
c7uc3u	u31	u27	u23	u19	u15	u11	u07	u03	c5sc1s	s29	s25	s21	s17	s13	s09	s05	s01
c7vc3v	v31	v27	v23	v19	v15	v11	v07	v03	c5tc1t	t29	t25	t21	t17	t13	t09	t05	t01
c4uc0u	u28	u24	u20	u16	u12	u08	u04	u00	c6sc2s	s30	s26	s22	s18	s14	s10	s06	s02
c4vc0v	v28	v24	v20	v16	v12	v08	v04	v00	c6tc2t	t30	t26	t22	t18	t14	t10	t06	t02
c5uc1u	u29	u25	u21	u17	u13	u09	u05	u01	c7sc3s	s31	s27	s23	s19	s15	s11	s07	s03
c5vc1v	v29	v25	v21	v17	v13	v09	v05	v01	c7tc3t	t31	t27	t23	t19	t15	t11	t07	t03

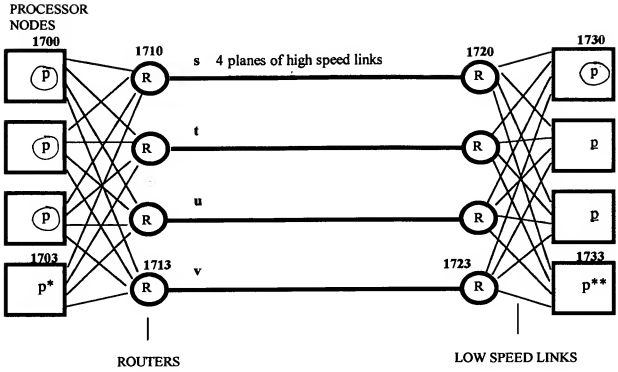
Data Flow >>> <<< NACK <<< ACK

FIG 14

Begin with Byte 00
Transmitted
s00t00 = 18h (data = ts = 81h) u00v00 = 77h (ECC = vu = 77h)
Received
s00t00 = 1Ah (data = ts = A1h) u00v00 = 75h (ECC = vu = 57h)
So correction proceeds exactly as before in Figure 8 for byte 00.
All 32 bytes are assembled and corrected then verified via the CRC checkcode comparison.

	ECC	Data
Data byte 00 input in error is		1 0 1 0 0 0 0 1 = A1 hex.
The ECC for A1 is F8	1 1 1 1 1 0 0 0 = F8 hex.	
ECC byte 00 input in error is	0 1 0 1 0 1 1 1 = 57 hex.	
The ECC syndrome	1 0 1 0 1 1 1 1 = AF hex.	
E.P. from Table 1 = d5 & e5	0 0 1 0 0 0 0 0 = e5	and 0 0 1 0 0 0 0 0 = d5
After corrections data = 81 hex.	0 1 1 1 0 1 1 1 = 77 hex.	and 1 0 0 0 0 0 0 1 = 81 hex.

FIG 14A



$[D_i]$ $d^0d^1d^2d^3d^4d^5d^6d^7$ = **st**

$[E_i]$ $e^0e^1e^2e^3e^4e^5e^6e^7$ = **uv**

$d^0d^1d^2d^3$ **s**

$d^4d^5d^6d^7$ **t**

$e^0e^1e^2e^3$ **u**

$e^4e^5e^6e^7$ **v**

su
sv **st** **st = [D_i]**
tu **uv**
tv
uv

data encoded with ECC

data dispersed across 4 planes

data assembly and recovery

data verification and correction

FIG 17